LAB OUTLINE

CE362 Signal Processing Lab
Pre-Requisite: NO Instructor: **Engr. Shehla Gul** Email**: shehla.gul@giki.edu.pk** Office # S-02, 2nd floor, New Academic Block Office Hours: 10:00 am ~ 05:00 pm

Lab Introduction

In this lab, we introduces the fundamental concepts in digital signals and its processing techniques. The objective of this lab is to introduce the processing of a discrete-domain information signal to enhance its quality and to extract important information using a digital processor. The processing of the signal could be done in the time domain or any other domain suitable for the application. General topics in the course include sampling and reconstruction of a signal, discrete-time signals and linear time-invariant systems, the z-Transform, discrete-time Fourier transform, fast Fourier transform, and discrete-time filters. These tools facilitate in building advanced applications such as speech recognition, speaker recognition, noise reduction, signal compression, signal classification, etc.

Mapping of CLOs and PLOs

Sr. No	Course Learning Outcomes+	PLOs*	Blooms Taxonomy
CLO_1	Upon successful completion of this course, the students will be able to describe and classify discrete time signals and systems in time domain as well as in frequency domain	P L O 1	C2 (Understanding)
CLO_2	Upon successful completion of this course, the students will be able to apply digital signal processing techniques to discrete time signals and systems	PLO 3	C3 (Applying)
CLO_3	Upon successful completion of this course, the students will be able to analyze and design discrete time systems/filters	PLO 2	C4 (Analysis)
	+Please add the prefix "Upon successful completion of this course, the student will be able to." *PLOs are for BS (CE) only		

CLO Assessment Mechanism (Tentative)

Assessment tools	CLO_1	CLO_2
Lab Performance	50%	50%
Open Ended Lab	-	15%
Midterm Exam	20%	-
Final Exam	30%	35%

Overall Grading Policy (Tentative)

Assessment Items	Percentage
Lab Performance	**40%**
Midterm Exam	**30%**
Open Ended Lab	**10%**
Final Exam	**20%**

Text and Reference Books

Text books:

- John G. Proakis and Dimitris G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, 4th edition, 2007.

Reference Material:

- MATLAB Signal Processing Tool
- K. Kim, Conceptual Digital Signal Processing with MATLAB, 1st Edition, 2021

Lab Breakdown

Lab	**Contents/Topics**
Lab#01:	Basics of MATLAB
Lab#02:	The effects of Sampling in Discrete Time Signals
Lab#03:	The effects of Quantization in Discrete Time Discrete Valued Signals.
Lab#04:	The impulse response, observe convolution technique in signal processing, and verify different properties like causality, commutative, distributive, and associative properties.
Lab#05:	The discrete time correlation and apply it to real data to observe the correlation between two signals.
Lab#06	The relationship between z-transform and DTFT
Mid Term Exam	
Lab#07:	To observe/find different frequency components in an audio signal
Lab#08:	Z transform and Inverse z transform
Lab#09:	Dft function-straighforward implementation and testing
Lab#10:	Exploring Properties of the Z-Transform
Lab#11:	Exploring Properties of the Z-Transform
Lab#12	**Open-Ended Lab**

	Final Exam

Evaluation Rubrics

	LAB RUBRICS	
Rubric 1	Understanding, Execution and Accuracy	5 marks
Rubric 2	Clarity and organization of the Code	3 marks
Rubric 3	Completion of Tasks and Adherence to the Instructions	2 marks

Detailed Rubric of Weekly Evaluation:

Rubric 1: Understanding, Execution, and Accuracy (5 marks)

This rubric assesses the students' comprehension of the lab concepts, their ability to execute tasks accurately, and the precision of their work.

- Understanding (2 marks): Evaluate how well students grasp the underlying concepts and theories relevant to the lab. Consider their ability to apply theoretical knowledge to practical tasks.
- Execution (2 marks): Assess how effectively students perform the assigned tasks. Look for their ability to follow instructions, use the required tools or software, and execute procedures accurately.
- Accuracy (1 mark): This includes the correctness of calculations, data analysis, and any other quantitative aspects of the lab. Consider the absence of errors in their execution.

Rubric 2: Clarity and organization of Code (3 marks)

This rubric evaluates the students' ability to code effectively and communicate their findings clearly.

- Evaluate the clarity and organization of the MATLAB code, including proper indentation, comments, and variable naming.

Rubric 3: Completion of Tasks and Adherence to the instructions (2 marks)

This rubric focuses on whether students have successfully completed all the assigned tasks within the lab.

- Task Completion (1 mark): Assess the extent to which students have completed the required tasks. This includes whether they have finished all the activities outlined in the lab manual or guidelines.
- Evaluate the extent to which the student followed instructions and guidelines provided in the lab manual.

Each rubric contributes to the overall evaluation of a student's performance in the lab, with a maximum total score of 10 marks. These rubrics provide a structured way to assess different aspects of the students' work and help ensure a fair and comprehensive evaluation.

Lab Manual 01

The effects of Sampling in Discrete Time Signals

Lab Objectives:

- To study the relationship between discrete-time and continuous time signals by examining sampling and aliasing.

Learning Outcomes:

Upon completing this lab, students will be able to:

- Define Sampling
- Explain Discrete-Time Signals
- Understand Sampling Rate
- Relate Sampling to Discrete-Time Signals
- Explore Aliasing
- Identify Aliased Signals

Signals:

Signals are physical quantities that carry information in their patterns of variation. Signal can be a function of time, distance, position, temperature, and pressure etc. for example, the traffic signals which changes with the time. So, the traffic signal is a function of time.

Continuous time vs Discrete time signals

Continuous time signals are continuous functions of time denoted by x(t), while discrete-time signals are sequences of numbers, denoted by x[n]. If the values of a sequence are chosen from a finite set of numbers, the sequence is known as a digital signal.

Continuous-time, continuous-amplitude signals are also known as analog signals. Analog phenomenon is continuous – like a human speech of speaker, or a continuously rotating disc attached to the shaft of motor etc. With analog phenomena, there is no clear separation between one point and the next; in fact, between any two points, an infinite number of other points exist.

Discrete phenomenon, on the other hand, is clearly separated. There's a point (in time or space), and then there's a neighboring point, and there's nothing between the two.

Signal Processing

Signal processing means analyzing the signal and performing operation on it. It is concerned with the acquisition, representation, manipulation, transformation, and extraction of information from signals.

In analog signal processing, these operations are implemented using analog electronic circuits. Converting the continuous phenomena of images, sound, and motion into a discrete representation that can be handled by a computer is called analog-to-digital conversion.

Digital signal processing

Digital signal processing involves the conversion of analog signals into digital, processing the obtained sequence of finite precision numbers using a digital signal processor or general purpose computer, and, if necessary, converting the resulting sequence back into analog form. When stored in a digital computer, the numbers are held in memory locations, so they would be indexed by memory address.

Regardless of the medium (either sound or an image), analog-to-digital conversion requires the same two steps:

Sampling and Quantization:

Sampling: it is phenomena of converting CTS to DTS.

This operation chooses discrete (finite) points at which to measure a continuous phenomenon (which we will also call a signal). In the case of sound, the sample points are evenly separated in time. In the case of images, the sample points are evenly separated in space.

Sampling Rate: The number of samples taken per unit time or unit space is called the sampling rate. The frequency of sampled/discrete phenomenon (signal) can be calculated as:

fd = F /Fs (cycles/sec)/(samples/sec) = cycles/ samples

Where, F = Frequency of analog or continuous phenomenon (signal). [Unit: cycles/sec]

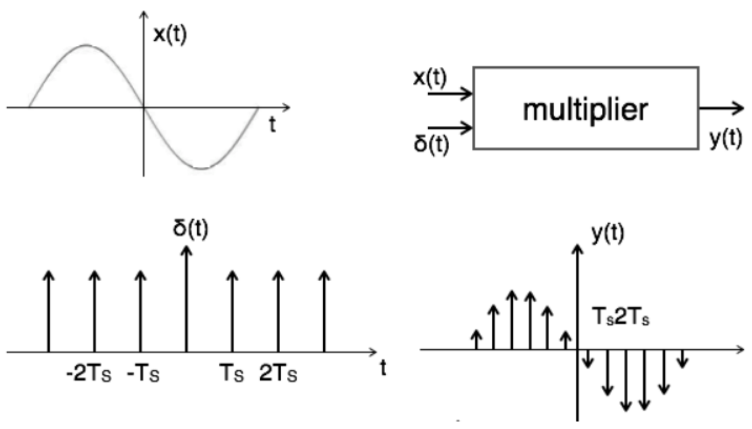
Fs = Sampling frequency or sampling rate [Unit: samples/sec]

fd = Normalized Frequency of Discrete phenomenon (signal). [Unit: cycles/sample]

Sampling Theorem:

A continuous time signal can be represented in its samples and can be recovered back when sampling frequency fs is greater than or equal to the twice the highest frequency component of message signal. i. e.

$$f_s \geq 2f_m.$$



Aliasing:

A common problem that arises when sampling a continuous signal is aliasing, where a sampled signal has replications of its sinusoidal components which can interfere with other components. It is an effect that causes two discrete time signals to become indistinct due to improper sampling. If the frequency of the continuous signal (F0) is greater than half of the sampling frequency (Fs/2), aliasing may occur or we can say that when sampling frequency is less than twice of input frequency signal, then it causes an aliasing effect.

$$F_s<2f_m \text{ or } F_0<F_s/2$$

PROCEDURE:

1. Simulate and plot two CT signals of 10 Hz and 110 Hz for $0 < t < 0.2$ secs.
2. Sample at Fs = 100 Hz and plot them in discrete form.
3. Observe and note the aliasing effects.
4. Explore and learn.

STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.

2. Open M-file editor and type the following code:

```
clear all;
close all;
clc;
F1 = 10;
F2 = 110;
Fs = 100;
Ts = 1/Fs;
 t = [0 : 0.0005 : 0.2];
x1t = cos(2*pi*F1*t);
x2t = cos(2*pi*F2*t);
figure, plot(t,x1t,t,x2t, 'LineWidth',2);
xlabel('cont time (sec)'); ylabel('Amp');
xlim([0 0.1]); grid on; legend('10Hz','110Hz');
title('Two CTCV sinusoids plotted');
```

3. Save the file as P011.m in your current directory and 'run' it, either using F5 key or writing the file name at the command window. (Check for the correctness of the time periods of both sinusoids.)
4. Now add the following bit of code at the bottom of your P011.m file and save.

```
nTs = [0 :Ts : 0.2]; %This line creates a vector nTs with values ranging from 0 to 0.2 with a step
size of Ts. The variable Ts represents the sampling interval.
n = [1 : length(nTs)-1 ]; %This line creates a vector n representing the discrete time indices. It
starts from 1 and goes up to the length of nTs minus 1.
x1n = cos(2*pi*F1*nTs); % This line generates a discrete-time signal x1n by sampling a cosine
wave with a frequency of F1 at the time instances given by nTs
x2n = cos(2*pi*F2*nTs);
figure, subplot(2,1,1), stem(nTs,x1n,'LineWidth',2);
grid on;
xlabel('discrete time (sec)');
ylabel('Amp'); xlim([0 0.1]);
subplot(2,1,2),
stem(nTs,x2n,'LineWidth',2);
grid on; title('110Hz sampled')
xlabel('discrete time(sec)');
 ylabel('Amp'); xlim([0 0.1]);
```

5. Before hitting the 'run', just try to understand what the code is doing and try to link it with what we have studied regarding concepts of frequency for DT signals.
6. Now 'run' the file and observe both plots. To see what is really happening, type the following code at the bottom of your existing P011.m file and run again.

```
figure, plot(t,x1t,t,x2t);
hold; stem(nTs,x1n,'r','LineWidth',2);
xlabel('time (sec)'); ylabel('Amp');
xlim([0 0.05]); legend('10Hz','110Hz');
```

7. Observe the plots.

RESULT:

Observe the cause and effects of what you just saw.

LAB TASKS:

1. **Consider the following CT signal:**

x(t) = sin (2 pi F0 t).

The sampled version will be x(n) = sin (2 pi F0/Fs n),

where n is a set of integers and sampling interval Ts=1/Fs.

Plot the signal x(n) for n = 0 to 99 for Fs = 5 kHz and F1 = 0.5, 2, 3 and 4.5 kHz.

Explain the similarities and differences among various plots.

2. **Generate a tone in MATLAB with varying frequency:**

f = 1000,2000,3000,4000, 5000, 6000, 8000, 9000, 25000, -1000, -2000, -3000 Hz with Fs = 8000 samples/sec. Listen to the tones, and observe at Sounds like what frequency? Also Specify whether Aliasing is happening or not.

Use the function ‘sound’ to generate tones of different frequencies.

Add a small pause between tones.

Lab Manual 02

The effects of Quantization in Discrete Time Discrete Valued Signals.

Lab Objectives:

- Understanding Quantization:
- Quantization Process:
- Effects of Quantization on Signal Quality:
- Quantization Error Analysis:

Learning Outcomes:

By the end of the lab, students should be able to:

- Define quantization in the context of discrete-time discrete-valued signals.
- Explain the process of quantization and its role in signal processing.
- Analyze the effects of quantization on the quality and fidelity of signals.
- Identify the trade-offs involved in choosing the number of quantization levels.
- Implement quantization in a programming environment.
- Observe and interpret the results of quantization on different types of signals.
- Conduct quantization error analysis and understand its impact on signal quality.

Everything stored on a computer is discrete time discrete valued signal. Because computer has finite number of registers, and each register is a finite length register. We take too many samples to give the 'effect' of continuous time signals. But actually, they are discrete time.

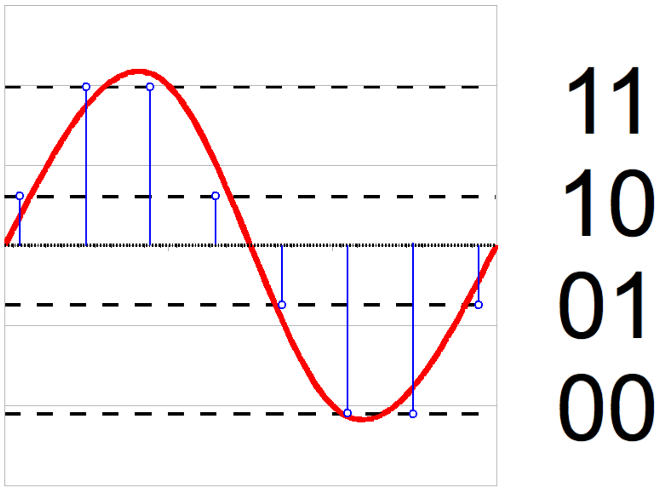
We also take very fine resolution of amplitude axis to give the effect of continuous valued signal but due to finite word length of the computer register, the stored variables are already quantized. This lab aims to explain the quantization effects in a computer.

Regardless of the medium (audio or image), the digitization of real world analog signal usually involves two stages:

Sampling, i.e. the measurement of signal at discretely spaced time intervals.

Quantization, i.e. the transformation of the measurements (amplitudes) into finite-precision numbers (allowed discrete levels), such that they can be represented in computer memory. Quantization is a matter of representing the amplitude of individual samples as integers expressed in binary. The fact that integers are used forces the samples to be measured in a finite number of bits **(discrete levels).**

Example:



Taking an example of this analog signal, whenever we need to do the quantization, we have to define the bit depth. The range of the integers possible is determined by the **bit depth**, the number of bits used per sample.

In the above example, if we consider n=2bits it means we are going to represent a sample by 2 bits which have been shown as 11, 10,01 and 00.

In digital sound, bit depth affects how much you have to round off the amplitude of the wave when it is sampled at various points in time.

Both sampling and quantization result in the loss of information. The quality of a Quantizer output depends upon the number of quantization levels used. The discrete amplitudes of the quantized output are called representation levels or reconstruction levels.

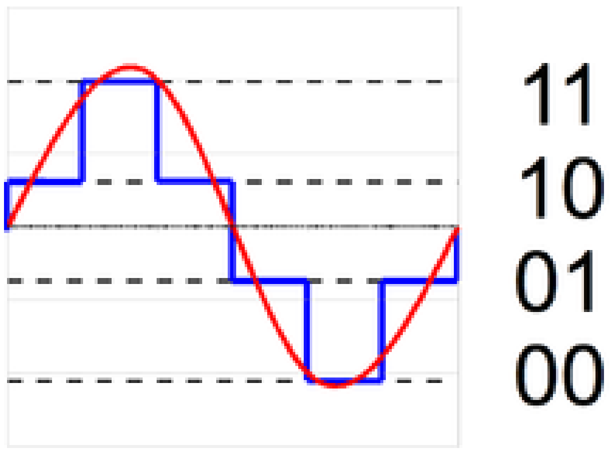
Now, the no of quantized levels L = 2 ^ n, where n is the bit depth. The L shows that we are defining the amplitude by 4 levels in case of bit depth = 2.

The spacing between the two adjacent representation levels is called **quantum or step-size.**

Step size Δ = X max – X min / L, where X max and X min are the maximum and minimum amplitude levels in the input signal.

Note: the bigger the step size, the higher the quantization error.

Below signal is the quantized signal according to the specified no of levels.



Quantization error:

The samples, which are taken at evenly spaced points in time, can take on the values only at discrete quantization levels to store on our computer. Therefore, quantization leads to a loss in the signal quality because it introduces a "Quantization error". Quantization error is sometimes referred to as '"Quantization noise". Noise can be broadly defined as part of an audio signal that isn't supposed to be there.

However, in terms of audio signal, a better term for quantization error is "distortion", defining distortion as an unwanted part of an audio signal that is related to the true signal. The difference between the quantized samples and the original samples constitutes quantization error or rounding error (if round-off method is used).

Xe(n) = Xq(n) − x(n).

The lower the bit depth, the more values potentially must be approximated (rounded), resulting in greater quantization error.

Practice Task:

1. Simulate a sinusoid of 1/50 cycles/sample with length of the signal be 500.
2. Choose the no. of significant digits for round-off and apply to the signal generated above.
3. Compute the error signals.

4. Explore and observe.

STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.
2. Open M-file editor and write the following code:

```
% Parameters
cycles_per_sample = 1/50; % Cycles per sample
signal_length = 500; % Length of the signal
significant_digits = 3; % No. of significant digits for round-off

% Generate DTCV sinusoid
t = (0:signal_length-1);
dtcv_signal = sin(2*pi*cycles_per_sample*t);

% Round-off to specified significant digits
rounded_signal = round(dtcv_signal, significant_digits);

% Compute error signal
error_signal = dtcv_signal - rounded_signal;

% Plotting
figure;
subplot(3,1,1);
plot(t, dtcv_signal);
title('Original DTCV Sinusoid');
xlabel('Sample');
ylabel('Amplitude');

subplot(3,1,2);
plot(t, rounded_signal);
title('Rounded Signal');
xlabel('Sample');
ylabel('Amplitude');

subplot(3,1,3);
plot(t, error_signal);
title('Error Signal');
xlabel('Sample');
ylabel('Amplitude');
```

3. Save the file as P021.m in your current directory and run it. Explore and observe the results.

LAB TASKS:

Effects of Quantization with variable precision levels

- Simulate a sampled composite signal of $fd1$=125 samples/sec and $fd2$=150 samples/sec with length of the signal be 250 samples.
- Take the desired number of significant digits from the user as input.
- Then choose the method of Quantization (round-off, floor & ceiling) and apply to the signal generated above.
- Compute the quantization error signals and explore the impact of each method.

Audio signal quantization to various bits per sample

- You need to record a 5-second audio clip and quantize it with only 1 significant digit.
- After plotting the original audio signal, the rounded audio signal, and the error signal, notice the results.
- Change bit depth to 2,3,4 and then listen and take notes of your observations.
- Decide no. of bits for audio until quality stops improving.
- Use audiorecorder function for recording the voice and soundsc function for playing the recorded audio.

Lab Manual 03

The impulse response, observe convolution technique in signal processing, and verify different properties like causality, commutative, distributive, and associative properties.

Lab Objectives:

- Understanding Impulse Response:
- Observing Convolution Technique:
- Verification of Properties:
- Causality Verification:
- Commutative Property Verification:
- Distributive Property Verification:
- Associative Property Verification:

Learning Outcomes:

By the end of the lab, students should be able to:

- Define impulse response and explain its importance in characterizing linear time-invariant systems.
- Apply convolution as a technique for combining signals and analyzing system behavior.
- Understand how convolution captures the response of a system to an input signal.
- Verify causality by examining the impulse response of a system.
- Analyze the causality property and its implications in real-world systems.
- Understand the commutative property and its significance in signal processing.
- Apply the distributive property to convolution operations.
- Appreciate the associative property and its applications in signal processing.

Introduction:

In linear time-invariant systems, breaking an input signal into individual time-shifted unit impulses allows the output to be expressed as the superposition of unit impulse responses. Convolution is the general method of calculating these output signals.

Convolution is an operation performed on two signals which involves multiplying one signal by a delayed or shifted version of another signal, integrating, or averaging the product, and repeating the process for different delays. Convolution is used to express the input and output relationship of an LTI system.

The convolution of two continuous-time signals x(t) and h(t) is represented as,

$$y(t)=x(t)*h(t)=\int \infty-\infty x(\tau)h(t-\tau)d\tau$$

Convolution of two discrete time signals is given as: y(n) = x(n)*h(n) =

$$\sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

i.e.one can compute the output y(n) to a certain input x(n) when impulse response h(n) of that system is known.

Causal signal is a **signal that has zero value for all negative time**.

While non-causal signals depend on past, present, and future values of time.

The limits of the integration in the convolution integral depends on whether the arbitrary signal x(t) and the impulse response h(t) are causal or not. Therefore,

- If both x(t)and h(t) are non-causal, then,
$$y(t)=\int x(\tau)h(t-\tau)d\tau y$$
- If the signal x(t) is non-causal and the impulse response h(t) is causal, then,
$$y(t)= {}_{-\infty}\int^{t} x(\tau)h(t-\tau)d\tau$$
- If the signal x(t) is causal and h(t) is non-causal, then
$$y(t)= {}_{0}\int^{\infty} x(\tau)h(t-\tau)d\tau$$
- If both x(t) and h(t) are causal, then,
$$y(t)= {}_{0}\int^{t} x(\tau)h(t-\tau)d\tau$$

Continuous-time convolution has basic , which are as follows –

- **Commutative Property of Convolution** – The commutative property of convolution states that the order in which we convolve two signals does not change the result, i.e.,

  $$x1(t)*x2(t)=x2(t)*x1(t)$$

- **Distributive Property of Convolution** –The distributive property of convolution states that if there are three signals x1(t), x2(t) an x3(t), then the convolution of x1(t) is distributive over the addition [x2(t)+x3(t)] i.e.,

  $$x1(t)*[x2(t)+x3(t)] = [x1(t)*x2(t)] +[x1(t)*x3(t)]$$

- **Associative Property of Convolution** – The associative property of convolution states that the way in which the signals are grouped in a convolution does not change the result, i.e.,

  $$x1(t)*[x2(t)*x3(t)] =[x1(t)*x2(t)]*x3(t)$$

PROCEDURE:

1. We have the impulse response of a system as h(n) = {3,2, 1, -2,1,0, -4,0,3}

2. For x(n) = {1, -2,3, -4,3,2,1}

STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.

2. Open M-file editor and write the following code:

```
clear all;
close all;
clc;
h = [3 2 1 -2 1 0 -4 0 3]; % impulse response
org_h = 1; % Sample number where origin exists
nh = [0 : length(h)-1]- org_h + 1; %
x = [1 -2 3 -4 3 2 1]; % input sequence
org_x = 1; % Sample number where origin exists
nx = [0 : length(x)-1]- org_x + 1;
y = conv(h,x);
ny = [nh(1)+ nx(1) : nh(end)+nx(end)];
figure,
subplot(3,1,1),
stem(nh,h);
xlabel('Time index n');
ylabel('Amplitude');
xlim([nh(1)-1 nh(end)+1]);
 title('Impulse Response h(n)');
grid;
```

```
subplot(3,1,2),
stem(nx,x);
xlabel('Time index n');
 ylabel('Amplitude');
xlim([nx(1)-1 nx(end)+1]);
title('Input Signal x(n)');
grid;
subplot(3,1,3)
stem(ny,y);
xlabel('Time index n');
ylabel('Amplitude');
xlim([ny(1)-1 ny(end)+1]);
title('Output Obtained by Convolution');
grid;
```

1. Save the file as P031.m in your current directory and 'run' it.

2. Calculate the length of input signal (N) and impulse response (M) used in above task?

3. Calculate the length of the output sequence and verify the result by using N+M-1.

4. Try to learn and explore the code.

5. Now modify the above code such that h(n)= {3,2, 1, -2,1,0, -4,0,3}(origin is shifted) and check for causality.

In the above code, nh = [0 : length(h)-1]- org_h + 1; defines a vector `h` that represents the impulse response of a linear system, and a scalar `org_h` that indicates the sample number where the origin exists. It then creates a vector `nh` that represents the time index of the impulse response, by subtracting `org_h` from the range of `0` to `length(h)-1`, and then adding `1`.

The code creates the vector nh as follows:
nh = [0 : length(h)-1]- org_h + 1;

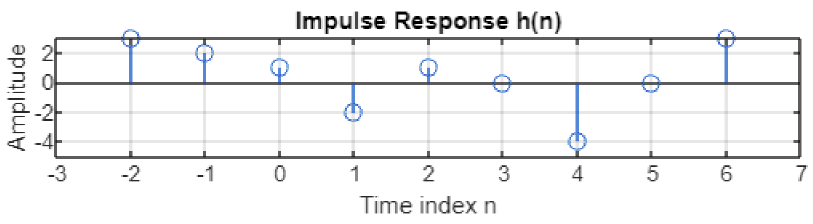
nh = [0 : 8]- 1 + 1; % length(h) is 9

nh = [0 1 2 3 4 5 6 7 8]- 1 + 1; % colon operator creates a vector from 0 to 8

nh = [-1 0 1 2 3 4 5 6 7] + 1; % subtraction operator subtracts 1 from each element

nh = [0 1 2 3 4 5 6 7 8]; % addition operator adds 1 to each element

The vector nh represents the time index of the impulse response h[n], such that h[0] = 3, h1 = 2, h2 = 1, and so on. The graph of the impulse response h[n] versus the time index n is shown below:



Lab Tasks:

1. What will happen if we input x(n) = {0,0,1,0,0} into the above system.
2. Can you prove the commutative property of the convolution?
3. Modify the code to prove Associative and Distributed properties of the convolution.

Lab Manual 04

The discrete time correlation and apply it to real data to observe the correlation between two signals.

Lab Objectives:

- Understanding Discrete Time Correlation:
- Correlation Types:
- Mathematical Formulation:
- Observing Real Data:
- Interpreting Correlation Results:
- Applications in Signal Processing:

Learning Outcomes:

By the end of the lab, students should be able to:

- Define discrete time correlation and its role in comparing signals.
- Differentiate between autocorrelation and cross-correlation.
- Formulate mathematical expressions for autocorrelation and cross-correlation.
- Apply discrete time correlation to real data sets.
- Analyze the results to identify patterns and similarities.
- Interpret correlation results in terms of signal behavior.
- Apply knowledge of discrete time correlation to practical signal processing scenarios.

Introduction:

The correlation of two functions or signals or waveforms is defined as the measure of similarity between those signals.

The correlation between two signals x(t) and y(t) is denoted as $r_{xy}(\tau)$ and is defined as:

$$r_{xy}(\tau)=\int_{-\infty}^{\infty} x(t)y(t+\tau)dt$$

In the discrete domain, for discrete-time signals x[n] and y[n], the correlation is defined as:

$$r_{xy}[k]=\sum x[n]y[n+k]$$

There are two types of correlations −

- Cross-correlation
- Autocorrelation

Cross-correlation

The cross-correlation between two different signals or functions or waveforms is defined as the measure of similarity or coherence between one signal and the time-delayed version of another signal. The cross-correlation between two different signals indicates the degree of relatedness between one signal and the time-delayed version of another signal.

The cross-correlation of energy (or aperiodic) signals is defined separately.

Cross-correlation of Energy Signals

Consider two complex signals x1(t) and x2(t)of finite energy. Then, the cross-correlation of these two energy signals is defined as

$$R12(\tau)= \int_{-\infty}^{\infty} x1(t)x2(t-\tau)dt=\int_{-\infty}^{\infty} x1(t+\tau)x2(t)dt$$

If the energy signals x1(t) and x2(t) have some similarity. Then, the cross-correlation R12(τ) between them will have some finite value over the range τ. The variable τ is called the *delay parameter or searching parameter or scanning parameter*. The time-delay parameter (τ) is the time delay or time shift of one of the two signals. This delay parameter τ determines the correlation between two signals.

Autocorrelation

The autocorrelation function is defined as the measure of similarity or coherence between a signal and its time delayed version. Therefore, the autocorrelation is the correlation of a signal with itself.

If the signal x(t) is shifted by τ units in negative direction, then the autocorrelation of the signal is defined as,

$$R11(\tau)=R(\tau)= \int_{-\infty}^{\infty} x(t+\tau)x*(t)dt$$

Relation between correlation and convolution:

- Cross-correlation is essentially convolution with one of the signals time-reversed:
- rxy(τ)=x(t)∗y(−t)
- If x(t)=y(−t), then correlation and convolution are the same operation.
- Convolution is commutative i.e., x(t)∗h(t)=h(t)∗x(t), while correlation is not commutative,
- rxy(τ)=ryx(τ).
- Convolution is used for filtering, system analysis, and solving integral equations, while correlation is used for pattern recognition, signal detection, and finding similarity between signals.

While correlation and convolution share mathematical similarities, they have different applications and interpretations in signal processing. Correlation measures similarity, while convolution combines signals, often representing system responses.

Practice Task:

1. Generate two sinusoids of length 10 and fd = 0.1 with variable phase.
2. Apply correlation and check for certain properties such as magnitude and location of maximum correlation with varying phases.

PROCEDURE:

1. Make a folder at desktop and name it as your current directory within MATLAB.

2. Open M-file editor and write the following code:

```
clear all;
close all;
clc;
n = [0:9];
ph1 = 0; %set the phase angles for the two sinusoidal signals to 0.
ph2 = 0;
x = sin(2*pi*0.1*n + ph1); % generates the first sinusoidal signal x with a frequency of 0.1 Hz and
phase ph1.
org_x = 1;
nx = [0 : length(x)-1]- org_x + 1;
y = sin(2*pi*0.1*n + ph2);
org_y = 1;
ny = [0 : length(y)-1]- org_y + 1;
rxy = xcorr(x,y);
nr = [nx(1)-ny(end) : nx(end)-ny(1)]; % defines the lag indices for the cross-correlation.lag is the
delay
[maxR indR] = max(rxy);
disp(['The correlation at lag zero is: ' num2str(rxy(find(nr==0))) '.']);
 disp(['The maximum correlation is at lag ' num2str(nr(indR)) '.']);
figure, subplot(3,1,1),
stem(nx,x);
xlabel('Time index n');
ylabel('Amplitude');
xlim([nx(1)-1 nx(end)+1]);
```

```
title('Signal x(n)');
grid; subplot(3,1,2),
stem(ny,y);
xlabel('Time index n');
ylabel('Amplitude');
xlim([ny(1)-1 ny(end)+1]);
title('Signal y(n)');
grid; subplot(3,1,3),
stem(nr,rxy);
xlabel('Time index n');
ylabel('Amplitude');
xlim([nr(1)-1 nr(end)+1]);
title('Cross Correlation');
grid;
```

Save the file as P041.m in your current directory and ‘run’ it.

It will show you following result.

Lab Tasks:

1. Now modify the phase of the second signal to pi/2 (it will make it cosine) and observe the correlation at lag zero.

2. Modify the phase again to ‘pi’ and observe.

3. Check for autocorrelation (ph1 = ph2) that the lag zero value gives the m energy of the signal.

4. Observe that the commutative property does not hold.

5. Modify the code, such that the correlation is obtained using convolution command.

Lab Manual 05

The relationship between z-transform and DTFT

Lab Objectives:

- To explore the mathematical relationship between the Z-Transform and DTFT.
- To learn methods for converting signals between the Z-Transform and DTFT domains.
- To gain practical experience in implementing these conversions using MATLAB.
- To analyze and interpret the frequency characteristics of signals and systems through magnitude and phase spectra.

Learning Outcomes:

By the end of this lab, students should be able to:

- Understand the mathematical relationship between the Z-Transform and DTFT and apply it to signal conversion.
- Implement algorithms to convert signals between the Z-Transform and DTFT domains using MATLAB.
- Interpret magnitude and phase spectra to analyze the frequency characteristics of signals and systems.
- Discuss the implications of the region of convergence (ROC) in Z-Transform analysis and system stability.
- Apply theoretical concepts to practical scenarios by generating signals, computing transforms, and visualizing frequency domain representation.

Discrete-Time Fourier Transform (DTFT):

The DTFT converts a time domain sequence into frequency domain signal. The DTFT of a discrete-time signal x[n] is defined as:

$$F[x(n)] = X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \quad \ldots(1)$$

The DTFT represents the spectrum of a discrete-time signal over the entire frequency range (-π to π).

Z-Transform:

Z-Transform is a mathematical tool used to analyze discrete-time signals and systems in the frequency domain. The Z-Transform converts a discrete-time signal from the time domain (n) to the complex frequency domain (z).

Mathematically, the Z-Transform of a discrete-time signal x[n] is defined as:

$$Z[x(n)] = X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad \ldots(2)$$

Relationship between Z-Transform and DTFT:

The Z-Transform and DTFT are related mathematically through the substitution of complex exponentials.

By substituting z = e^(jω) in the Z-Transform, we obtain the relationship between the Z-Transform and DTFT:

This relationship illustrates that the DTFT is a specific instance of the Z-Transform when z is replaced by e^(jω).

Therefore, it can be said that the Z-transform of a discrete time sequence x(n) is same as the discrete-time Fourier transform (DTFT) of x(n)r^-n, i.e.,

$$Z[x(n)] = F[x(n)r^{-n}]$$

Magnitude Spectrum:

The magnitude spectrum represents the magnitude (or amplitude) of each frequency component present in the signal. It shows how much of each frequency is present in the signal. In the case of the Z-Transform and DTFT, comparing the magnitude spectra obtained from both transformations helps to verify if the frequency content of the signal is preserved accurately during the transformation process.

Phase Spectrum:

The phase spectrum represents the phase shift of each frequency component present in the signal. It shows how the phase of each frequency component changes with respect to the reference. In the case of the Z-Transform and DTFT, comparing the phase spectra obtained from both transformations helps to ensure that the phase relationships between different frequency components are maintained accurately during the transformation process.

The relationship between the Z-Transform and DTFT is established based on the substitution z=e^jω. This relationship allows us to relate the frequency domain representation obtained from the Z-Transform to that obtained from the DTFT. By comparing the magnitude and phase spectra obtained from both transformations, we can verify if the frequency content and phase relationships of the signal are consistent between the two domains. This verification is essential for ensuring the accuracy and validity of signal analysis and processing techniques in both the time and frequency domains.

Practice Task: Verifying the Relationship between Z-Transform and DTFT:

```
% Generate a discrete-time signal x[n]
n = 0:50; % Time index
x = sin(0.1*pi*n) + 0.5*sin(0.2*pi*n); % Example signal (sum of two sinusoids)

% Plot input signal x[n]
figure;
subplot(3,1,1);
stem(n, x, 'b', 'LineWidth', 2);
xlabel('n');
ylabel('x[n]');
title('Input Signal');

% Compute the Z-Transform of the input signal x[n] using the definition
syms z n0; % Define symbolic variables
X_z = sum(x .* z.^(-n)); % Z-Transform calculation

% Plot Z-Transform of the input signal
subplot(3,1,2);
ezplot(abs(X_z), [-2, 2]);
xlabel('Re(z)');
ylabel('|X(z)|');
```

```
title('Magnitude of Z-Transform');

% Compute the DTFT of the input signal x[n]
omega = -pi:0.01:pi; % Frequency range
X_dtft = zeros(size(omega)); % Initialize DTFT
for k = 1:length(omega)
   X_dtft(k) = sum(x .* exp(-1j * omega(k) * n)); % DTFT formula
end

% Plot magnitude spectra
subplot(3,1,3);
plot(omega, abs(X_dtft), 'b', 'LineWidth', 2);
hold on;
plot(omega, abs(subs(X_z, z, exp(1j*omega))), 'r--', 'LineWidth', 2);
xlabel('\omega (radians/sample)');
ylabel('Magnitude');
title('Magnitude Spectrum Comparison');
legend('DTFT', 'Z-Transform (z = e^{j\omega})');
grid on;

% Adjust plot settings
set(gcf, 'Position', [100, 100, 800, 800]);
```

Lab task:

Given a Z-Transform expression, Perform the substitution z=ejω to convert the Z-Transform into the DTFT. Write MATLAB code to implement the conversion:

- Define the symbolic variable for frequency, ω.
- Substitute ejω for z in the Z-Transform expression.
- Compute the DTFT expression.
- Plot the magnitude and phase spectra of both the original Z-Transform and the converted DTFT expressions for comparison.
- Analyze the frequency characteristics of the signal represented by both transformations and compare the results.
- Use a substitute ejω for z in the Z-Transform expression to obtain the DTFT expression X_dtft.

Lab Manual 06

To observe/find different frequency components in an audio signal

Lab Objectives:

- Understanding Fourier Analysis
- Application of Discrete Fourier Transform (DFT)
- Frequency Spectrum Analysis
- Visualization Techniques

This lab aims to introduce students to the concept of frequency analysis in signal processing using Fourier Transforms. Starting with the basic concepts,

Audio Signal: An audio signal is a representation of sound waves captured over time. In digital form, it's typically a series of discrete samples representing the amplitude of the sound wave at different time intervals.

Fourier Transform: The Fourier Transform is a mathematical technique used to decompose a signal into its constituent frequencies. It converts a signal from the time domain (amplitude vs. time) to the frequency domain (amplitude vs. frequency). This transformation allows us to analyze the frequency content of a signal.

Frequency Components: In the frequency domain, a signal is represented by its frequency components, each corresponding to a specific frequency and amplitude. By analyzing these components, we can identify the dominant frequencies present in the signal.

Plotting Frequency Spectrum: The frequency spectrum is a graphical representation of the frequency components of a signal. It typically shows the amplitude of each frequency.

DF Analysis Equations:

The DF analysis equations represent the process of decomposing a discrete signal into its frequency components using the Discrete Fourier Transform (DFT).

DFT Synthesis Equation:

The DFT synthesis equation is the inverse operation of the DFT analysis equations.

It reconstructs the original discrete signal from its frequency components obtained through the DFT.

Frequency Resolution:

Frequency resolution refers to the ability to distinguish between closely spaced frequency components in the frequency domain. In the context of the DFT, frequency resolution is determined by the sampling frequency and the length of the signal. Higher frequency resolution allows for more precise identification of frequency components.

Sample Frequencies:

Sample frequency (or sampling rate) refers to the number of samples taken per unit time during the digitization of an analog signal. It determines the highest frequency that can be accurately represented in the digital signal, according to the Nyquist-Shannon sampling theorem.

The choice of sample frequency affects the accuracy and frequency resolution of the Fourier analysis.

Selecting an appropriate sample frequency is crucial to avoid aliasing (incorrect representation of high-frequency components) and to ensure accurate frequency analysis.

In the lab, you will apply these concepts to analyze the frequency content of audio signals using the DFT. You'll use DF analysis equations to decompose the signal into its frequency components, synthesize the signal using the DFT synthesis equation, consider frequency resolution to accurately identify frequency components, and select suitable sample frequencies to ensure accurate representation of the signal.

DF analysis equation: $X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0,1,\ldots,N-1$

DFT synthesis eq: $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}, \quad n = 0,1,\ldots,N-1$

Frequency Resolution is given by $\Delta F = \frac{F_s}{N}$.

Sample frequencies are given by $F_k = \Delta F \times k, \; k = 0,1,\ldots,N-1$.

Fundamental Energy:

The energy of the fundamental frequency and its harmonics provides information about the strength and prominence of these frequency components in the signal. Higher energy levels indicate stronger presence and potentially clearer perception of these frequencies. It is calculated by squaring the magnitude of fundamental frequency component.

Lab Task:

- Perform a frequency analysis of any recorded sound signal by analyzing its time and frequency domains.
- Calculate the DFT.
- Plot the sound signal in the time domain and its Discrete Fourier Transform (DFT) magnitude spectrum in the frequency domain.
- Identify the fundamental frequency.
- Calculate the energy present in both the fundamental frequency and its harmonics.

Lab Manual 07

Z transform and Inverse z transform

Lab Objectives:

- To manipulate the z, transform and inverse z transform practically in the MATLAB

The Z-transform is an essential tool in digital signal processing (DSP) for analyzing and processing discrete-time signals. It is a mathematical transformation that maps a discrete-time signal in the time domain to a complex-valued function in the frequency domain.

The Z-transform of a discrete-time signal x(n) is defined as the infinite sum of the signal values weighted by powers of a complex variable z. $X(z)=\sum_{n=-\infty}^{\infty} x(n)z^{-n}$

Computing the Z-transform in MATLAB

MATLAB provides a convenient way to compute the Z-transform of a discrete-time signal using the ztrans function. The syntax of the function is as follows:

```
F = ztrans(f, n, z)
```

where f is the symbolic expression of the discrete-time signal, n is the symbolic variable representing the time index, and z is the symbolic variable representing the complex frequency variable. The output F is the symbolic expression of the Z-transform of the signal.

For example, let's consider discrete sine signal, then the following code:

```
syms n z

f = sin(n);

F = ztrans(f, n, z);
```

Here, we define a symbolic variable n to represent the time index and a symbolic variable z to represent the complex frequency variable. We then define the signal f as sin(n). Finally, we use the ztrans function to compute the Z-transform of the signal and store it in the variable F.

Plotting the Z-transform in MATLAB

Once we have computed the Z-transform of a signal, we can plot its magnitude and phase as a function of the normalized frequency using MATLAB's plotting functions. In this example, we will use the fplot function to plot the magnitude and phase of the Z-transform of the signal.

To plot the magnitude of the Z-transform, we can use the following code:

```
fplot(abs(F), [-pi, pi]);

title('Magnitude of F(z)');
```

```
xlabel('Frequency (radians)');

ylabel('|F(z)|');
```

Here, we use the fplot function to plot the absolute value (abs) of the Z-transform F over the frequency range from -π to π. We then set the title and axis labels of the plot.

Similarly, to plot the phase of the Z-transform, we can use the following code:

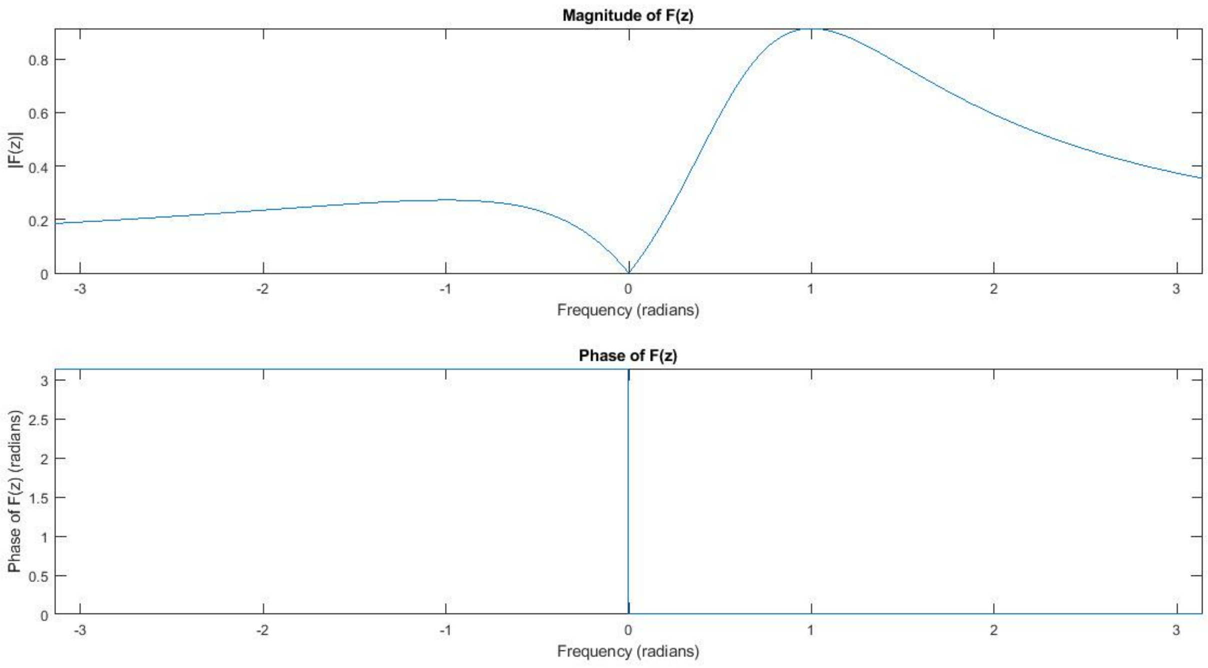
```
fplot(angle(F), [-pi, pi]);

title('Phase of F(z)');

xlabel('Frequency (radians)');

ylabel('Phase of F(z) (radians)');
```

Here, we use the fplot function to plot the phase (angle) of the Z-transform F over the same frequency range. We then set the title and axis labels of the plot.

Complete MATLAB Z-Transform code

```
clear

syms n z

f = sin(n);

F = ztrans(f, n, z);

% Plot the magnitude of F(z)

fplot(abs(F), [-pi, pi]);

title('Magnitude of F(z)');

xlabel('Frequency (radians)');

ylabel('|F(z)|');

% Plot the phase of F(z)

fplot(angle(F), [-pi, pi]);

title('Phase of F(z)');

xlabel('Frequency (radians)');

ylabel('Phase of F(z) (radians)');
```

MATLAB Z-Transform Plot



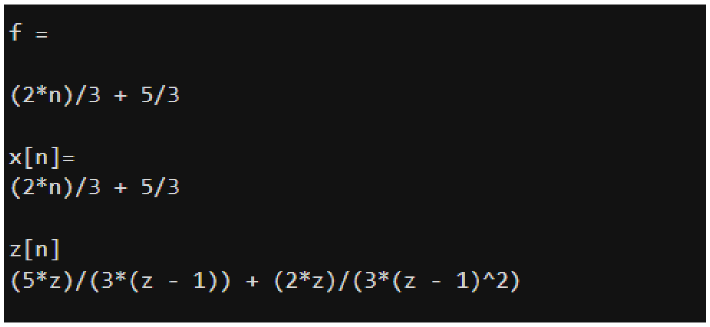
Solving equations with the z transform:

Here is a simple example of an equation that also shows some little details.

Code:

```
syms n;

f=(2*n+5)/3

disp('x[n]=')

disp(f)

F=ztrans(f)

disp('z[n]')

disp(F)
```

Output:



```
f =

(2*n)/3 + 5/3

x[n]=
(2*n)/3 + 5/3

z[n]
(5*z)/(3*(z - 1)) + (2*z)/(3*(z - 1)^2)
```

Similarly, for the inverse z transform, we need to use the MATLAB function of inverse z transform. To apply the inverse z transform in MATLAB, we use the following formula:

iztrans(x)

Lab tasks:

1. Write MATLAB code for the z transform of function 0.5^n?
2. Write MATLAB code for the z transform of function sin(n)?
3. Write MATLAB code for the inverse z transform of function 2*z/(z-2)^2?
4. Find z transform of following:
    - n^2
    - n*a^2
    - a^n
    - n*((0.5)^n*cos9pi*n/3))
5. Find inverse z transform of (2*z)/(2*z-1)

Lab Manual 08

DFT FUNCTION-STRAIGHFORWARD IMPLEMENTATION AND TESTING

Lab objectives:

- Introduce students to the theoretical background of the Discrete Fourier Transform (DFT), including its mathematical formulation and its significance in signal processing.
- Guide students through the implementation of the DFT algorithm in MATLAB.
- Visualize the input signal, its DFT spectrum, and the reconstructed signal from the inverse DFT.

Discrete Fourier Transform:

The discrete-time Fourier transform (DTFT) or the Fourier transform of a discrete–time sequence x[n] is a representation of the sequence in terms of the complex exponential sequence e^$j\omega n$.

The formulas for the DFT and IDFT are given as

$$X(K) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \;; \qquad k=0,1,\ldots\ldots N-1$$

$$X(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k)\, W_N^{-kn} \;; \qquad k=0,1,\ldots\ldots N-1$$

Where by definition $W_N = e^{\frac{-j2\Pi}{N}}$

Which is an Nth root of unity. Where W_N is called Twiddle Factor , also

$$[W_N] = [e^{-j2\Pi/N}]^{kn}$$

$$W_N^{kn} = e^{-j2\Pi kn/N}$$

DFT analysis equation in matrix form is

$$X_N = [\,W_N^{kn}\,]x_N$$

DFT synthesis equation in matrix form is

$$x_N = [W_N^{kn}]^{-1} X_N$$

Practice Task:

Compute 4-point DFT of x(n)= (1,2,3,0).

STEPS

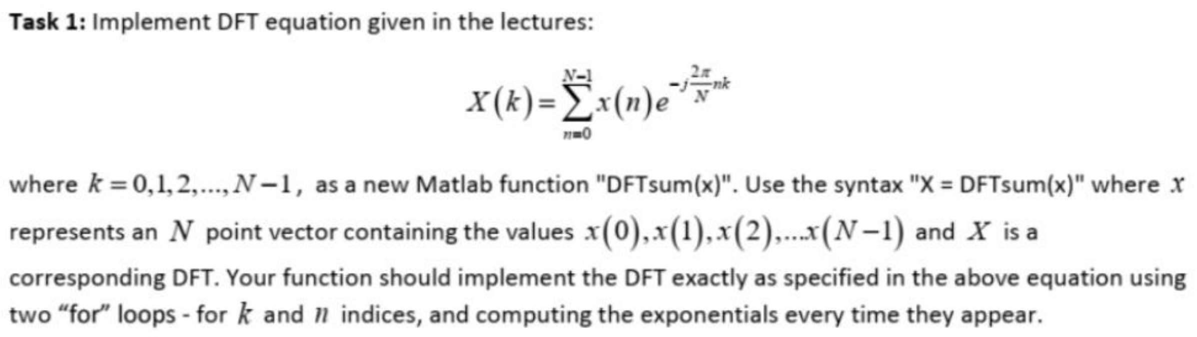
- Generate given sequence in MATLAB.

- Take N-=4 to calculate 4-point DFT.
- Define 0: N-1 point vector for time and frequency samples.
- Define W matrix and then use DFT analysis equation to compute DFT.

Code:

```
x = [1, 2, 3, 0]; % Define the input sequence

N = 4;      % Length of the sequence

n = [0:1:N-1];  % Time index vector

k = [0:1:N-1];   % Frequency index vector

% Compute the DFT using matrix multiplication

WN = exp (-1i * 2 * pi / N); % Twiddle factor

nk = n' * k;        % Create a matrix of indices

WNnk = WN.^ nk;          % Compute the twiddle factor matrix

Xk = x * WNnk;   % Compute the DFT
```

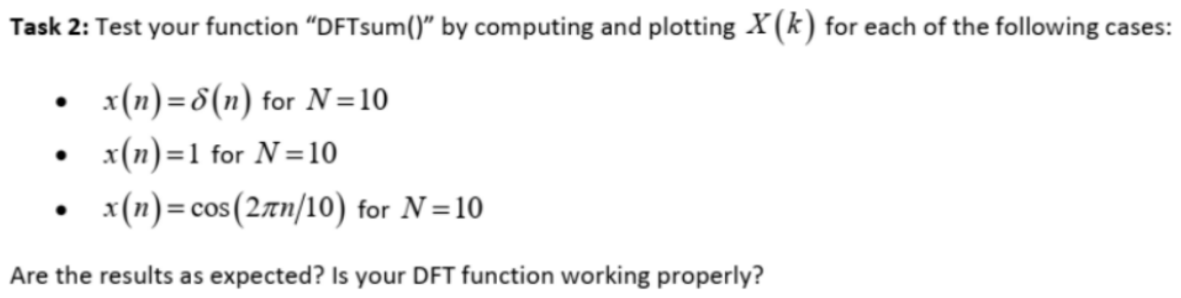
Lab Tasks:



Task 1: Implement DFT equation given in the lectures:

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi}{N}nk}$$

where $k=0,1,2,\ldots,N-1$, as a new Matlab function "DFTsum(x)". Use the syntax "X = DFTsum(x)" where x represents an N point vector containing the values $x(0),x(1),x(2),\ldots x(N-1)$ and X is a corresponding DFT. Your function should implement the DFT exactly as specified in the above equation using two "for" loops - for k and n indices, and computing the exponentials every time they appear.



Task 2: Test your function "DFTsum()" by computing and plotting $X(k)$ for each of the following cases:

- $x(n)=\delta(n)$ for $N=10$
- $x(n)=1$ for $N=10$
- $x(n)=\cos(2\pi n/10)$ for $N=10$

Are the results as expected? Is your DFT function working properly?

Task 3: Write a second Matlab function for computing the inverse DFT according to the IDFT equation:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi}{N}nk}$$

Use the syntax "x = IDFTsum(X)" where $X(k)$ is the N point vector containing the DFT result and $x(n)$ represents the corresponding time-domain signal.

Task 4: Use "IDFTsum(X)" to invert each of the DFT's computed in Task 2. Plot the magnitudes of the inverted DFT's, and verify that the obtained time-domain signals match the original ones.

Note: You may need to take abs(X) or real(X) to eliminate any imaginary parts which roundoff errors during IDTFT calculations may produce.

Lab Manual 09

Exploring Properties of the Z-Transform

Lab objectives:

- Gain a clear understanding of fundamental properties such as linearity, time shifting, convolution, and scaling properties of the Z-transform.
- Learn how to perform Z-transform operations in MATLAB to analyze discrete-time signals and systems.
- Apply theoretical concepts of the Z-transform to practical scenarios by verifying properties using MATLAB simulations.
- Enhance programming skills in MATLAB by implementing algorithms to compute Z-transforms and verify properties.

Introduction:

The Z-transform is a powerful tool used in digital signal processing to analyze discrete-time signals and systems. It provides a way to transform a discrete-time signal from the time domain to the Z-domain, where operations such as filtering and analysis become simpler.

In this lab, we will explore some fundamental properties of the Z-transform using MATLAB. Before starting the lab tasks, make sure you have a basic understanding of the Z-transform and its properties.

Properties of the Z-Transform:

1. **Linearity Property:**

The linearity property of the Z-transform states that for any two signals $x1[n]$ and x2[n] and any constants a and b, the Z-transform of their linear combination $ax1[n]+bx2[n]$ is equal to the linear combination of their individual Z-transforms. Mathematically, this can be expressed as:

$Z\{ax1[n]+bx2[n]\}=aX1(z)+bX2(z$

Where $X1(z)$ and $X2(z)$ are the Z-transforms of $x1[n]$ and $x2[n]$ respectively.

2. **Time Shifting Property:**

The time shifting property of the Z-transform states that if a discrete-time signal $x[n]$ is shifted in time by k samples to obtain $x[n-k]$, then the Z-transform of the shifted signal $x[n-k]$ is given by $z-kX(z)$, where $X(z)$ is the Z-transform of $x[n]$. In other words, a time shift in the time domain results in a multiplication by a corresponding power of z in the Z-domain.

3. **Convolution Property:**

Explanation: The convolution property of the Z-transform states that the Z-transform of the convolution of two signals $x[n]$ and $h[n]$ is equal to the product of their individual Z-transforms. Mathematically, this can be expressed as:

$Y(z)=X(z)\cdot H(z)$

Where $X(z)$ and $H(z)$ are the Z-transforms of $x[n]$ and $h[n]$ respectively, and $Y(z)$ is the Z-transform of their convolution $y[n]$.

4. **Time Reversal Property:**

The time reversal property of the Z-transform states that if a discrete-time signal $x[n]$ is reversed in time to obtain $x[-n]$, then the Z-transform of the reversed signal $x[-n]$ is given by $X(1/z$, where $X(z)$ is the Z-transform of $x[n]$. In other words, time reversal in the time domain corresponds to inversion in the Z-domain.

5. **Scaling Property:**

Explanation: The scaling property of the Z-transform states that if a discrete-time signal $x[n]$ is scaled by a constant c to obtain $cx[n]$, then the Z-transform of the scaled signal $cx[n]$ is given by $X(cz)$, where $X(z)$ is the Z-transform of $x[n]$. In other words, scaling a signal in the time domain by a constant factor results in the multiplication of its Z-transform by the same constant factor in the Z-domain.

Mathematically, this can be expressed as:

$Z\{cx[n]\}=X(cz)$

Where $X(z)$is the Z-transform of $x[n]$.

Example:

Let's consider a simple example to illustrate the scaling property. Suppose we have a discrete-time signal $x[n]=\{1,2,3\}$, and we want to scale it by a factor of $c=2$ to obtain $2x[n]=\{2,4,6\}$

The Z-transform of $x[n]$ is given by:

$X(z)=1+2z-1+3z-2$

Using the scaling property, the Z-transform of $2x[n]$ would be:

$Z\{2x[n]\}=X(2z)=1+2(2z)-1+3(2z)-2$

$=1+22z+34z2$

$=1+1/z+3/4z2$

This illustrates how the Z-transform of the scaled signal $2x[n]$ is obtained by replacing z with 2z in the Z3.

6. Initial Value Theorem:

The initial value theorem of the Z-transform states that the value of a discrete-time signal $x[n]$ at $n=0$ is equal to the limit of $X(z)$ as z approaches infinity. Mathematically, this can be expressed as:

$x[0]=\lim z\to\infty X(z)$

Where $X(z)$ is the Z-transform of $x[n]$

-transform expression of x[n].

Lab Tasks:

Task 1:

- Generate a discrete-time signal $x[n]$ with known values.
- Shift the signal in time by a certain number of samples (positive or negative).
- Compute the Z-transforms of the original signal $x[n]$ and the shifted signal.
- Verify the time shifting property of the Z-transform by comparing the Z-transforms of the original and shifted signals.

Task 2:

- Create two discrete-time signals $x[n]$ and $h[n]$ with known values.
- Convolve the signals in the time domain to obtain the output signal $y[n]$.
- Compute the Z-transforms of $x[n]$, $h[n]$, and $y[n]$.
- Verify the convolution property of the Z-transform by confirming that the Z-transform of $y[n]$ is equal to the product of the Z-transforms of $x[n]$ and $h[n]$.